Exhibit B

PROCEEDINGS 1998

DESIGN AUTOMATION CONFERENCE®

Moscone Center, San Francisco, CA
June 15 - 19, 1998

Sponsored by

EDA CONSORTIUM

sig da

acm

PROCEEDING OF THE 35th DESIGN AUTOMATION CONFERENCE

The Association for Computing Machinery, Inc.
1515 Broadway
New York, NY 10036

ACM Order Number 477980
ISBN 0-89791-964-5

Library of Congress Number 85-644924

IEEE Catalog Number 98CH36175
ISBN 0-7803-4409-X (Casebound Edition)
ISBN 0-7803-4410-3 (Microfiche Edition)

ISSN 0738-100X

Additional copies of 1998 or prior Proceedings may be ordered prepaid from

ACM Order Department
P.O. Box 11414
Church Street Station
New York, NY 10286

ACM European Service Center
108 Cowley Rd.
Oxford, OX41JF, U.K.
Phone: 44-1865-382338
Fax: 44-1865-381338
E-mail: ACM_Europe@acm.org

Phone: 1-800-342-6626
(U.S.A. and Canada)
+1-212-626-0500
(All other countries)
Fax: +1-212-944-1318
E-mail: acmpubs@acm.org

Additional copies of this publication are available from

IEEE Service Center
P.O. Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331

1-800-678-IEEE
1-908-981-1393
1-908-981-9667 (Fax)
833-233 (Telex)

Printed in the U.S.A.

Table of Contents

Executive Plenary Panel: Customers, Vendors, and Universities: Determining the Future of EDA Together

Session 1
Interfaces for Design Reuse

Session 2
Analog and Mixed-Signal Design Tools

Session 3
University Design Contest

Chair: Mary Jane Irwin

Organizer: Jan M. Rabaey

Session 4
Embedded System Design and Exploration

Chair: Ivo Bolsens

Organizers: James A. Rowson, Anders Forsen

Session 5
Taming Noise in Deep-Submicron Digital Designs

Chair: Nagaraj NS

Organizers: Kenneth L. Shepard, Nagaraj NS

Panel: Taming Noise in Deep Submicron Digital Integrated Circuits

Chair: Nagaraj NS

Organizers: Kenneth Shepard, Takahide Inoue

Session 6
Control and Data Driven High Level Synthesis

Chair: Kayhan Kucukcakar
Organizers: David Ku, Timothy Kam

Session 7
Synthesis Flow in Deep Submicron Technologies

Chair: Ralph H. J. M. Otten
Organizers: Sharad Malik, Randal E. Bryant

Session 8
Environment for Collaborative Design

Chair: Takahide Inoue
Organizers: Richard Smith, Takahide Inoue

Session 9
New Methods in Functional Verification

Chair: Rajesh K. Gupta
Organizers: Vivek Tiwari, Kenji Yoshida

Session 10
Panel: Hardware/Software Co-Design: The Next Embedded System Design Challenge

Session 11
System-Level Power Optimization

Session 12
Boolean Methods

Session 13
Extraction and Modeling for Interconnect

Session 14
Processor Design and Simulation

Session 15
Panel: How Much Analog Does a Designer Need To Know for Successful Mixed-Signal Design?

Session 16
Performance Modeling and Characterization for Embedded Systems

Session 17
Advances in Placement and Partitioning

Session 18
Parasitic Device Extraction and Interconnect Modeling

Session 19
Design Optimization for DSP

Session 20
Panel: User Experience with High Level Formal Verification

Session 21
Bridging the Gap Between Simulation and Formal Verification

Session 22
Logic Optimization

Session 23
Routing for Performance and Crosstalk

Chair: Sachin S. Sapatnekar

Organizers: Patrick Groenveld, Andrew B. Kahng

Session 24
Practical Optimization Methodologies for High Performance Design

Chair: Vivek Tiwari

Organizers: Vivek Tiwari, Kenji Yoshida

Session 25
RF IC Design Methodology

Chair: Mojy C. Chian

Organizers: Bryan D. Ackland, Mojy C. Chian

Session 26
Theory and Practice in High Level Synthesis

Chair: Steve Tjiang

Organizers: David Ku, Timothy Kam

Session 27
BDD Approximation Techniques

Session 28
Interconnect Modeling and Timing Simulation

Session 29
Low Power Design Using Multiple Thresholds and Supplies

Session 30
Panel: Technical Challenges of IP and System-on-Chip: The ASIC Vendor Perspective

Session 31
Software Synthesis and Retargetable Compilation

Session 32
Formal Methods in Functional Verification

Session 33
Core Test and BIST

Session 34
Interconnect Analysis and Reliability in Deep Sub-Micron

Session 35
Panel: Design Productivity: How to Measure It, How to Improve It

Session 36
Timing Analysis

Session 37
New Techniques in State Space Explorations

Session 38
Advanced ATPG Techniques

Session 39
Practical Experience of Functional Verification for Complex ICs

Chair: Rajesh Raina

Organizers: David T. Blaauw, Kenji Yoshida

Session 40
Panel: The EDA Start-up Experience: The First Product

Chair: Erach Desai

Organizer: Mike Murray

Session 41
Fast Functional Simulation

Chair: Patrick C. McGeer

Organizers: Rajesh K. Gupta, David Ku

Session 42
Power Estimation and Modeling

Chair: Farid N. Najm

Organizers: Farid N. Najm, Andrew T. Yang

Session 43
Technology Mapping for Programmable Logic

Chair: Jonathan Rose

Organizers: Jason Cong, TingTing Hwang

Session 44
Power Dissipation and Distribution in High Performance Processors

Chair: David T. Blaauw

Organizers: Anatha Chandrakasan, Jan M. Rabaey

Session 45
Challenge in the Test on System-On-A-Chip Era

Chair: Prab Varma

Organizers: Prab Varma, Takahide Inoue

Session 46
Controller Decomposition for Power and Area Minimization

Chair: Fabio Somenzi

Organizers: Timothy Kam, Luciano Lavagno

Session 47
IP Protection Technologies

Chair: Tom VandenBerge
Organizers: Richard Smith, Takahide Inoue

Session 48
Case Studies of New Design Methods

Chair: Anders Forsen
Organizers: Ivo Bolsens, James A. Rowson

Generic Global Placement and Floorplanning

Hans Eisenmann and Frank M. Johannes

http://www.regent.e-technik.tu-muenchen.de

Institute of Electronic Design Automation
Technical University Munich
80290 Munich
Germany

Abstract

We present a new force directed method for global placement. Besides the well-known wire length dependent forces we use additional forces to reduce cell overlaps and to consider the placement area. Compared to existing approaches, the main advantage is that the algorithm provides increased flexibility and enables a variety of demanding applications. Our algorithm is capable of addressing the problems of global placement, floorplanning, timing minimization and interaction to logic synthesis. Among the considered objective functions are area, timing, congestion and heat distribution. The iterative nature of the algorithm assures that timing requirements are precisely met. While showing similar CPU time requirements it outperforms Gordian by an average of 6 percent and TimberWolf by an average of 8 percent in wire length and yields significantly better timing results.

1 Introduction

Automated cell placement for VLSI circuits has always been a key factor for achieving designs with optimized area usage and timing behavior. Beyond this, the deep-submicron era is posing new challenges onto a placement tool: Meeting timing specifications is becoming more difficult, floorplanning requires that larger designs are placed in less time and a variety of additional physical and geometrical constraints must be fulfilled simultaneously.

A common formulation of the placement objective is to minimize wire length under the constraint that cells don't overlap each other. Early formulations of the placement problem use forces for reducing the overlaps between cells [1]. Complexity and convergence problems quickly led to the development of more powerful methods. The current state-of-the-art placement tools for handling large designs can be classified into two categories based on how they make sure that the placement is free of overlaps. The first class consists of algorithms which keep the placement free of overlaps during the whole placement process. Among them, the simulated annealing method showed excellent results [2]. The second class of algorithms is based on a hierarchical subdivision of the placement area with a corresponding partitioning of the set of cells. Within this approach, a min-cut objective [3, 4] has been used successfully and the combination with a quadratic objective function showed best results [5, 6, 7]. Partitioning based methods typically yield a placement with small overlaps between cells, a so-called global placement, which has to be made free of overlaps by a final placement step.

Decreasing feature sizes require that timing issues are considered during layout synthesis. Among various techniques to reduce circuit delays, placement algorithms can be extended to be timing driven. Timing driven placement aims at minimizing the wire length of nets along critical paths. Net based approaches transform path constraints to net constraints or net weights [8] whereas path based approaches can consider the length of the longest path directly [9].

The mixed-block/cell placement occurs in floorplanning applications. It is typically converted into a block placement problem by assigning cells to flexible blocks. Flexible block placement can be solved efficiently [10]. Alternatively, the locations of the blocks are set constant before or during the placement process.

We present a force directed method which uses a new approach of dealing with cell overlaps. We use the well-known force directed formulation and apply additional forces to reduce cell overlaps and to distribute cells over the placement area. Our iterative approach has the advantage that no hard constraints are used during the placement procedure. This property makes our approach much more powerful than partitioning based methods which make irreversible decisions at early stages based on incomplete or inaccurate information. Moreover, avoiding hard constraints gives the algorithm the flexibility to address a variety of applications and objective functions. For example, our algorithm is the first one which is able to handle large mixed block/cell placement problems without treating blocks and cells differently. Contrary to existing state-of-the-art methods, our approach can assure that timing requirements are precisely met.

The rest of the paper is organized as follows: In section 2, we formulate the placement task using the well-known wire length dependent forces and add an additional force to each cell. This formulation transforms the placement problem into the problem of finding the additional forces. Section 3 addresses the problem of determining the additional forces to reduce cell overlaps and to adapt the placement to the placement area. The results from section 2 and 3 are used in section 4 to formulate an iterative algorithm. Section 5 illustrates applications for generic cell placement, floorplanning, timing minimization and interaction with logic synthesis. In section 6, comparisons with state-of-the-art algorithms demonstrate the superiority of our approach with regard to wire length and timing optimization.

DAC 98, June 15-19, 1998, San Francisco, CA USA
ISBN 1-58113-049-x/98/06…$5.00

2 Problem Formulation

In this section, we begin with the standard formulation of the placement problem where a quadratic objective function is used to model wire length. We introduce additional forces working on each cell. This formulation transforms the problem of finding cell coordinates into the problem of finding forces. The benefit of our formulation is that it inherently incorporates wire length minimization. The problem of determining additional forces is addressed in section 3.

2.1 Quadratic Objective Function and Additional Forces

Let n be the number of movable cells in the circuit, x_i the x-coordinate of the center of cell i and y_i the y-coordinate of the center of cell i. A placement of the circuit can be described by the $2n$-dimensional vector $\vec{p} = (x_1,\ldots,x_i,\ldots,x_n,y_1,\ldots,y_i,\ldots,y_n)^T$.

As usual, we are modeling the circuit connectivity as a graph. Cells are modeled as vertices and nets are modeled as edges. A net connecting k cells yields a clique in the graph. The clique consists of $k(k-1)/2$ edges with weight $1/k$ connecting each vertex with all other vertices.

We formulate the cost of an edge as the squared Euclidean distance between its adjacent vertices multiplied with the weight of the edge. The squared Euclidean distance between two movable cells i and j is $(x_i - x_j)^2 + (y_i - y_j)^2$. If cell i is connected to a fixed cell with coordinates (x_f, y_f), the distance is $(x_i - x_f)^2 + (y_i - y_f)^2$. Consequently, the objective function sums up the cost of all edges and can be written in matrix notation [11]

$$\frac{1}{2}\vec{p}^T C \vec{p} + \vec{d}^T\vec{p} + const \qquad (1)$$

by using the $2n \times 2n$ symmetric matrix C and the $2n$-dimensional vector $\vec{d}$. For example, the x-part of the connection between two movable cells i and j is $(x_i - x_j)^2 = x_i^2 - 2\cdot x_i \cdot x_j + x_j^2$. The first term contributes to the diagonal of C at row i, the second term causes negative entries at row i, column j and at row j, column i. The third term is a contribution to the diagonal of C at row j. In case of a fixed connection, $(x_i - x_f)^2$ evaluates to $x_i^2 - 2\cdot x_i \cdot x_f + x_f^2$. The first term is a contribution to the diagonal of C, the second term gives a negative entry at $\vec{d}$ in row i and the third term affects the constant part of (1). The objective function (1) is minimized by solving the linear equation system

$$C\vec{p} + \vec{d} = 0 \qquad (2)$$

This formulation is equivalent to modeling nets as springs and calculating the state of equilibrium. In detail, row i (row $i+n$) of equation system (2) states that the force working on cell i is zero in x direction (y direction).

2.2 Additional Forces

In the following, we introduce additional forces working on each cell. We extend equation (2) with the force vector $\vec{e}$ to model constant additional forces:

$$C\vec{p} + \vec{d} + \vec{e} = 0 \qquad (3)$$

The force vector $\vec{e}$ contains the additional forces working on the cells in the x and y direction. Equation (3) transforms the problem of finding a placement into the problem of finding additional forces $\vec{e}$ and calculating the placement $\vec{p}$ according to (3). It is easy to show that the introduction of forces does not restrict the solution space, i.e. any given placement can fulfill equation (3) if the additional forces $\vec{e}$ are chosen appropriately.

3 Determination of the Additional Forces

In this section, we present a specific choice of how additional forces can be used. We take advantage from the fact that equation (2) already inherently minimizes wire length and formulate a specific choice for the additional forces from which we derive a unique set of forces.

3.1 Specific Choice of Additional Forces

Additional forces shall be used to distribute the cells evenly over the placement area.

Motivation: Solving equation (2) gives the global optimum with regard to squared wire length. However, equation (2) neither considers the overlaps of the cells nor the placement area. Therefore, the resulting placement is overlapping and not well distributed over the placement area in general. We use the additional forces in equation (3) to remove cell overlaps and to adapt the placement to the placement area.

3.2 Requirements

In the following, we derive four requirements for the additional forces.

1. For a given placement, the additional force working on a cell depends only on the coordinates of the cell.

   Motivation: If the task of the forces is to reduce the overlaps of the cells, it is only natural that a cell at the same place as another cell gets the same force.

2. Regions with higher density are the sources of the forces. Regions with lower density are the sinks.

   Motivation: Forces should move cells away from high density regions and lead them to low density regions.

3. The forces do not form circles.

   Motivation: The purpose of the forces is to improve the distribution. A circular force does not improve the distribution and is of no use.

4. In infinity, the force should be zero.

   Motivation: This requirement prevents the existence of a constant force.

3.3 Mathematical Formulation and Solution

We now show that these four requirements are sufficient to uniquely determine the forces. The basic background of how the requirements can be put into mathematical formulations can be found in [12].

The first requirement allows us to write the additional force $\vec{f_i}$ at cell i as a function of x and y: $\vec{f_i} = \vec{f}(x,y)|_{x=x_i,y=y_i}$

For the second requirement we define a supply-and-demand model for describing the density at a given point. We first define the rectangle function $R(z)$:

$$R(z) = \begin{cases} 1 & \text{if } -1/2 < z < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

The width of cell i is denoted by w_i, the height of cell i is h_i. The function $a_i(x,y)$ describes the area of cell i and is defined as $a_i(x,y) = R(\frac{x-x_i}{w_i}) \cdot R(\frac{y-y_i}{h_i})$. The value of $a_i(x,y)$ is one for a point (x,y) which is covered by cell i and zero otherwise.

Using W as the width of the placement area and H as its height, the area function $A(x,y)$ of the placement area can be defined similarly as $A(x,y) = R(\frac{x}{W}) \cdot R(\frac{y}{H})$. The value of $A(x,y)$ is one for a point (x,y) within the placement area and zero otherwise.

With the definition of s as the quotient of total cell area and placement area we define a function for the density $D(x,y)$:

$$D(x,y) = \sum_i a_i(x,y) - s \cdot A(x,y) \;\text{ with }\; s = \frac{\sum_i w_i \cdot h_i}{W \cdot H} \tag{4}$$

In other words: The density at a certain point within the placement area is the number of cells which cover the point, minus s.

The first term of equation (4) can be seen as the demand of area by the cells. The second term is the supply of area from the placement area scaled by factor s. The integral of $D(x,y)$ over the whole area is zero. This is achieved by scaling the supply with s. Let us note that $D(x,y) > 0$ at locations with higher density than desired and $D(x,y) < 0$ at locations with lower density than desired.

Using a proportional constant k, requirement 2 gives [12]

$$\vec{\nabla} \vec{f}(x,y) = k \cdot D(x,y) \tag{5}$$

Requirement 3 means that $\vec{f}(x,y)$ is conservative, i.e. there exists a scalar function $\Phi(x,y)$ with

$$\vec{\nabla}\Phi(x,y) = \vec{f}(x,y) \tag{6}$$

Combining (5) and (6) results in Poisson's equation

$$\Delta\Phi(x,y) = k \cdot D(x,y) \tag{7}$$

with boundary conditions from requirement 4

$$\lim_{|\vec{r}|\to\infty} |\vec{\nabla}\Phi(x,y)| = 0 \;\text{ with }\; \vec{r} = (x,y)^T \tag{8}$$

This standard problem has a unique solution for $\vec{f}(x,y)$ [13] :

$$\vec{f}(x,y) = \frac{k}{2\pi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} D(x',y') \frac{\vec{r}-\vec{r}'}{|\vec{r}-\vec{r}'|^2} dx'\, dy' \tag{9}$$

3.4 Interpretation

In the case of modeling cells as points and subdividing the placement area into places with unit area, the above integral becomes a discrete sum and equation (9) can be interpreted informally to make its meaning and its nature more clear:

1. The force working on a cell is the superposition of the forces originating from other cells and places.
2. The force exerted on a cell by another cell (place) is repelling (attracting) with a strength proportional to the inverse of their Euclidean distance.
3. The direction of the force is parallel to the straight line between the cells (between the cell and the place, resp.).

4 Basic Algorithm

We now combine the results from section 2 and 3. The forces in section 2 have been assumed to be constant whereas section 3 states that forces depend on the location of the corresponding cell. We resolve this conflict by introducing an iterative algorithm which determines the forces according to the current placement. These forces are set constant and used to calculate a new placement. The new placement then is the base for the next iteration step and so on.

We first describe a single iteration step called 'placement transformation'. Then, we show how a global placement can be achieved by successively applying placement transformations.

4.1 Placement Transformation

We call one step of the algorithm a placement transformation. The input placement can be arbitrary. The transformation step can be applied to fully overlapping placements as well as nearly legal placements. We do not consider information from the past like the number of iterations or a certain cooling schedule.

We determine the forces $\vec{f}_i$ of the current placement according to equation (9). We choose the proportional constant k so that the maximum strength of all forces $\vec{f}_i$ is equivalent to the force of a net with length $K \cdot (W + H)$ with a constant parameter K as user parameter. The parameter K determines how strongly the additional forces influence the placement and therefore determines the speed of convergence and the quality of the results. Next, we add the determined additional forces $\vec{f}_i$ to the force vector $\vec{e}$: $(\Delta e_i, \Delta e_{n+i})^T = \vec{f}_i$. Matrix C is set up as described in section 2.1. We apply a linearization scheme for adjusting netweights [14] and solve equation (3) by using a conjugate gradient approach with preconditioning. During the solving step of equation (3) we assume that the additional forces $\vec{e}$ remain constant.

4.2 The Iterative Algorithm

The complete algorithm consists of three key elements:

1. Initialization: In the initial routine, all cells are placed at the center of the placement area and the forces $\vec{e}$ are set to zero.
2. Iteration loop: We iteratively apply placement transformations. We use K to control the desired speed of the algorithm. We choose $K = 0.2$ for standard behavior while $K = 1.0$ is used for fast operation. Each iteration makes the distribution of the cells more even and adapts the placement more to the placement area.
3. Stopping criterion: We stop the iterations when there exists no empty square within the placement area which is larger than four times the average area of a cell. Our experiments showed that this criterion is sufficient for a desired even distribution of cells.

5 Application to Placement Tasks

In this section we describe what placement tasks can be addressed by the algorithm. We also describe which extensions have to be made for the different applications.

Timing Optimization We apply an iterative net weighting approach in order to optimize the timing behavior. We use the Elmore delay model based on the half perimeter of the enclosing rectangle as net delay. At iteration step m, each net j has a criticality $c_j^{(m)}$. We initialize $c_j^{(0)}$ with zero and update it as follows.

Before each placement transformation, we carry out a longest path search for timing analysis. This gives us the maximum delay and the minimum slack for each net. Please note that our approach does not rely on this special model as it works well with any kind of timing analysis.

We define the criticality of net j at step m as

$$c_j^{(m)} = \begin{cases} (c_j^{(m-1)} + 1)/2 & \text{if net } j \text{ is among the 3 percent most} \\ & \text{critical nets} \\ c_j^{(m-1)}/2 & \text{otherwise} \end{cases}$$

The criticality describes how critical a net tends to be in general and is used later for our weight updating scheme. A net which is critical at step m contributes 50 % to its criticality, a net which is critical at step $m-1$ contributes 25 % and so on. For example, if a net was never critical its criticality is zero whereas an always critical net has a criticality of one. This scheme effectively reduces oscillations of netweights.

Each net j has an associated weight $w_j^{(m)}$. The initial weights $w_j^{(0)}$ are set to one. We multiply the weight $w_j^{(m-1)}$ of net j with $1+c_j^{(m)}$ and get the new weight $w_j^{(m)}$. For example, a net which has never been critical has $c_j^{(m)}=0$ and keeps its weight $w_j^{(m)} = w_j^{(m-1)}$. The weight of a net which has always been critical is multiplied by a factor of 2: $w_j^{(m)} = 2 \cdot w_j^{(m-1)}$.

Our timing optimization especially benefits from the fact that even in late stages the placement has the ability to change globally because there are no hard constraints. This property makes our approach much more powerful than other methods where decisions at early stages have to be made based on incomplete or inaccurate information.

Meeting Timing Requirements The task of meeting a given timing requirement is different from optimizing the longest path delay. A timing requirement is a constraint under which other criteria should be optimized.

The typical approach with existing methods is iterative on a coarse level. The timing behavior of the resulting placement is compared to the specifications. Then, the input data is changed to make the method more aggressive or less aggressive. Additional placement runs are carried out until the desired timing behavior is obtained.

In order to avoid these multiple placement runs, we propose an extension to the described algorithm which directly meets timing requirements. First, we run the basic non-timing driven algorithm until it converges. We now have an area optimized placement. Then, we carry out a net weight adaption before each placement transformation. This net weight adaption is the same as described for timing optimization. In order to get a timing/area tradeoff curve we record the wire length and timing data for each step. We stop after the timing requirements are met. Since we used the resulting placement for timing analysis we can assure that the placement meets precisely the timing requirements.

Compared to existing methods, our approach has two major advantages: It guarantees that the timing requirements are precisely met if it is possible at all. No further reruns of the placement algorithm are necessary. Secondly, our two-phase approach provides a tradeoff curve showing which timing can be achieved at which area cost.

Mixed Block Placement and Floorplanning Using placement programs for floorplanning can yield problem sizes of several hundreds of thousands of cells. Floorplanning is a highly interactive task and the placement step is carried out multiple times resulting in the need for fast placement. Moreover, floorplanning poses the problem of placing big blocks and small cells simultaneously. Existing approaches are either not able to cope with the large problem size or disregard the dimension of the big blocks (at least at some step of the algorithm). Simulated annealing methods suffer from the fact that moving big cells is computationally intensive and partitioning based methods disregard the dimension of the big cells.

Contrary to that, our algorithm is the first one which is able to handle large mixed block/cell placement problems without treating blocks and cells differently.

Congestion and Heat Driven Placement Since the modeling of the force sources is done by a supply and demand model, it is straightforward to extend the procedure to incorporate congestion information. Before each placement transformation a routing estimation is executed. Then, a congestion map is determined which is used in combination with the density $D(x,y)$ to calculate additional forces. With this approach, the placement and the congestion map converge simultaneously and yield a placement which directly considers the resulting congestion. Additionally, by replacing the congestion map with a heat map we can use the same approach to avoid hot spots in the layout.

ECO and Interaction with Logic Synthesis Our approach is well suited for netlist changes during placement and for considering netlist changes after placement. This feature is typically needed by the application of ECO, gate resizing techniques and fully automated interaction with logic synthesis. The main requirement is that the existing placement is disturbed as little as possible.

Our method starts from the given placement and introduces additional forces according to the density deviations arising from netlist changes. Any changes in the netlist result in additional forces which move the surroundings slightly in order to adapt to the changed situation. The placement of cells relative to each other is preserved. The deviations in density are typically small which leads to small additional forces resulting in small changes for the placement. Compared to other methods, our approach has the advantage that an incrementally changed netlist results in small changes in the placement.

6 Experimental Results

We used the benchmark set from [15] for comparison. Table 1 shows the parameters of the circuits. Wire length is measured by summing up the half perimeter of the enclosing rectangle for each net. CPU times are measured in seconds for an Alphastation 250/4-266. CPU times of other approaches are scaled according to [16].

6.1 Standard Cell Benchmarks

We compare our results to the state-of-the-art methods Gordian/Domino [14, 17] and TimberWolf [2]. As final placer for the proposed method we used Domino [17]. The reported CPU times include the time used for final placement. The Gordian/Domino values are taken from [2]. We take the TimberWolf values from [18, 19] which are summarized in [2].

Since results for circuit *struct* have not been published, we ran TimberWolf 1.3.0 and Gordian/Domino 9.4 in their default configuration to obtain the values for circuit *struct*.

Table 1 lists the absolute values. The columns ‘wire length’ show the wire length in meters and ‘CPU’ the CPU time in seconds. We used the standard mode ($K=0.2$) of our approach.

For better comparison, we also list the results relative to other methods in table 2. The columns ‘improvement’ give the wire length improvement in percent. Positive values mean that our approach is better. Columns ‘rel. CPU’ give the CPU time of our approach divided by the (scaled) time of the compared approach. Therefore, values smaller than 1.0 mean that our approach is faster.

One can see that our results are comparable with TimberWolf while using one third of the runtime. A comparison under similar runtime conditions (approximately 40 percent slower) outperforms TimberWolf by 7.9 percent and Gordian/Domino by 6.6 percent in wire length on the average.

We further investigated the quality/runtime tradeoff of our approach. Our motivation is that a fast placement is useful to achieve a placement estimation during the floorplanning phase. We compared the fast ($K=1.0$) and the standard ($K=0.2$) mode of our

circuit	# cells	#nets	# rows	T.-Wolf [19]		T.-Wolf [18]		Go./Do. [17]		Our Approach	
				wire length [m]	CPU [s]	wire length [m]	CPU [s]	wire length [m]	CPU [s]	wire length [m]	CPU [s]
fract	125	147	6								
primary1	752	904	16	-	-	0.99	22	0.88	17	0.87	37
struct	1888	1920	21	0.364	272	-	-	0.377	30	0.338	40
primary2	2907	3029	28	-	-	3.72	125	3.68	92	3.72	152
biomed	6417	5742	46	1.62	885	1.88	216	1.95	264	1.78	284
industry2	12142	13419	72	13.53	3285	14.34	925	15.80	958	14.6	1283
industry3	15059	21940	54	42.84	4504	44.67	877	44.97	1034	45.1	1605
avq.small	21854	22124	80	5.41	4587	6.13	1302	5.68	1744	4.91	1741
avq.large	25114	25384	86	5.86	5501	6.81	1560	6.21	2108	5.38	2031

Table 1: Benchmarks: Wire Length and CPU Time

circuit	T.-Wolf [19]		T.-Wolf [18]		Go./Do. [17]	
	% improvement	rel. CPU	% improvement	rel. CPU	% improvement	rel. CPU
primary1	-	-	+12.1	1.68	+1.1	2.17
struct	+7.1	0.14	-	-	+11.5	1.33
primary2	-	-	0.0	1.21	-1.1	1.65
biomed	-9.8	0.32	+5.3	1.31	+8.7	1.07
industry2	-7.9	0.39	-1.8	1.38	+7.5	1.33
industry3	-5.2	0.35	-0.9	1.83	-0.2	1.55
avq.small	+9.2	0.37	+19.9	1.33	+13.5	0.99
avq.large	+8.1	0.36	+20.9	1.30	+13.3	0.96
average	+0.2	0.32	+7.9	1.43	+6.6	1.38

Table 2: Comparisons to Other Approaches: Wire Length Improvement and Relative CPU Times

approach. Using the fast mode, we can calculate a placement in approximately one third of the time compared to the standard mode. The average wire length increase is 6 percent. Within 10 minutes, the fast approach is capable of obtaining a legal placement for 25 000 cells which could not be obtained by any other approach.

6.2 Timing

For better comparison to existing methods, we introduce a lower bound for the length of the longest path by setting all wire lengths to zero and performing a timing analysis. This lower bound can only be reached if all nets of the longest path have length zero which means that all cells would be interconnected by abutment. Then, we calculate the longest path of a placement without timing optimization and subtract the lower bound. This gives the optimization potential of the placement. Next, we calculate the reduction of the longest path by using timing optimization. Dividing this value by the optimization potential tells us how much the method could exploit the optimization potential, i.e. how good it is. Measuring the performance in the proposed way helps us to reduce the influence of differences in net models, timing models and timing parameters.

We compare our method to [20] and [21] which has been shown to be superior to [22]. Since having big nets in the longest path is not realistic we disregard nets with more than 60 pins for timing analysis in our approach. Therefore, the lower bound used for our approach is different from the lower bound used for the other approaches in the case of ‘avq.small’ and ‘avq.large’. Using the same lower bound would give unrealistic good results.

As in [21], we assume a capacitance per length of 242 pF/m and a resistance per length of 25.5 kΩ/m. Table 3 shows the results. Columns ‘without timing’ list the results without timing optimization, ‘with timing’ the results for timing optimization. All timing values are in nanoseconds.

Table 4 demonstrates the capability of exploiting the optimization potential. The columns ‘lower bound’ list the lower bound in nanoseconds which we used for calculating the optimization potential. The columns ‘exploitation’ show the exploitation of the optimization potential. Higher values mean better timing optimization. Columns ‘relative CPU’ contain the CPU time requirement relative to our approach. Values larger than one mean that the compared method is slower.

On the average, the compared methods can exploit the optimization potential by up to 42 percent whereas our method utilizes 53 percent of the optimization potential and requires less CPU time.

7 Conclusions

The upcoming placement tasks pose new challenges to placement tools. They must be capable of placing larger netlist in less time, of considering more geometrical and physical constraints, and of dealing with more complicated timing requirements. Moreover, they must provide a flexible interface for manual changes as well as for changes introduced by logic synthesis tools.

To address these tasks, we proposed an iterative force directed method. A new approach has been presented which uses additional forces to remove cell overlaps and to adapt the placement to the placement area. The approach has the advantage that no hard constraints are used during the placement procedure. This property makes our approach much more powerful than existing methods which make irreversible decisions at early stages that are based on incomplete or inaccurate information. The avoidance of hard constraints gives the algorithm the flexibility to address a variety of applications and objective functions. For example, our algorithm is the first one which is able to handle large mixed block/cell placement problems without treating blocks and cells differently. Contrary to state-of-the-art approaches, our approach can assure that timing requirements are precisely met. While using comparable or less CPU time, our approach outperforms Gordian and TimberWolf by an average of 6 and 8 percent in wire length and yields significantly better timing results.

8 Acknowledgments

The authors thank the reviewers for their valuable suggestions. The first author thanks Kurt Antreich for his continuous support.

	TimberWolf [20]			Speed [21]			Our Approach		
	without	with	CPU	without	with	CPU	without	with	CPU
circuit	timing	timing		timing	timing		timing	timing	
	[ns]	[ns]	[s]	[ns]	[ns]	[s]	[ns]	[ns]	[s]
fract	208	131	500	-	-	-	21.2	19.8	3
struct	907	449	1450	108	101	32	92.5	90.1	50
biomed	-	-	-	126	65	218	48.6	35.7	397
avq.small	1106	798	31993	827	580	3792	102	80	2791
avq.large	-	-	-	836	605	7185	113	94	3642

Table 3: Timing Results: Longest Path and CPU Time

	TimberWolf [20]			Speed [21]			Our Approach		
	lower	exploi-	rel.	lower	exploi-	rel.	lower	exploi-	rel.
circuit	bound [ns]	tation	CPU	bound [ns]	tation	CPU	bound [ns]	tation	CPU
fract	18.5	40%	164	-	-	-	18.5	51%	1
struct	84.0	55%	28	84.0	29%	0.64	84.0	28%	1
biomed	-	-	-	27.0	61%	0.55	27.0	59%	1
avq.small	142	31%	11	142	36%	1.35	69.9	68%	1
avq.large	-	-	-	142	33%	1.97	79.9	57%	1
average		42%	67		40%	1.13		53%	1

Table 4: Relative Timing Results: Exploitation of Optimization Potential and relative CPU requirements

References

[1] N. Quinn and M. Breuer, “A force directed component placement procedure for printed circuit boards,” *IEEE Trans. CAS*, vol. CAS-26, pp. 377–388, June 1979.

[2] W.-J. Sun and C. Sechen, “Efficient and effective placement for very large circuits,” *IEEE Trans. CAD*, vol. 14, no. 3, pp. 349–359, 1995.

[3] A. Dunlop and B. Kernighan, “A procedure for placement of standard-cell VLSI circuits,” *IEEE Trans. CAD*, vol. CAD-4, pp. 92–98, Jan. 1985.

[4] D. J.-H. Huang and A. B. Kahng, “Partitioning based standard cell global placement with an exact objective,” in *ISPD*, pp. 18–25, 1997.

[5] C.-K. Cheng and E. S. Kuh, “Module placement based on resistive network optimization,” *IEEE Trans. CAD*, vol. CAD-3, pp. 218–225, July 1984.

[6] R.-S. Tsay, E. S. Kuh, and C.-P. Hsu, “PROUD: A fast sea-of-gates placement algorithm,” in *ACM/IEEE DAC*, vol. 25, pp. 318–323, 1988.

[7] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, “GORDIAN: VLSI placement by quadratic programming and slicing optimization,” *IEEE Trans. CAD*, vol. CAD-10, pp. 356–365, Mar. 1991.

[8] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, “Chip layout optimization using critical path weighting,” in *ACM/IEEE DAC*, vol. 21, pp. 133–136, 1984.

[9] M. A. B. Jackson and E. S. Kuh, “Performance-driven placement of cell based IC’s,” in *ACM/IEEE DAC*, vol. 26, pp. 370–375, 1989.

[10] R. Otten, “Efficient floorplan optimization,” in *IEEE ICCD*, pp. 499–501, Oct. 1983.

[11] K. M. Hall, “An r-dimensional quadratic placement algorithm,” *Management Science*, vol. 17, pp. 219–229, Nov. 1970.

[12] R. Ellis and D. Gulick, *Calculus*. Harcourt Brace Jovanovich, 1991.

[13] W. E. Williams, *Partial Differential Equations*. Oxford University Press, 1980.

[14] G. Sigl, K. Doll, and F. M. Johannes, “Analytical placement: A linear or a quadratic objective function?,” in *ACM/IEEE DAC*, 1991.

[15] “www.cbl.ncsu.edu/benchmarks/layoutsynth92/”

[16] “performance.netlib.org/performance/html/PDStop.html”

[17] K. Doll, F. M. Johannes, and K. J. Antreich, “Iterative placement improvement by network flow methods,” *IEEE Trans. CAD*, vol. 13, pp. 1190–1200, Oct. 1994.

[18] W.-J. Sun and C. Sechen, “Efficient and effective placement for very large circuits,” in *IEEE/ACMICCAD*, pp. 170–177, 1993.

[19] W.-J. Sun and C. Sechen, “A loosely coupled parallel algorithm for standard cell placement,” in *IEEE/ACMICCAD*, pp. 137–144, 1994.

[20] W. Swartz and C. Sechen, “Timing driven placement for large standard cell circuits,” in *ACM/IEEE DAC*, 1995.

[21] B. M. Riess and G. G. Ettelt, “Speed: Fast and efficient timing driven placement,” in *IEEE ISCAS*, 1995.

[22] A. Srinivasan, K. Chaudhary, and E. S. Kuh, “RITUAL: A performance driven placement algorithm,” *IEEE Trans. CAS*, vol. CAS-39, pp. 825–840, Nov. 1992.